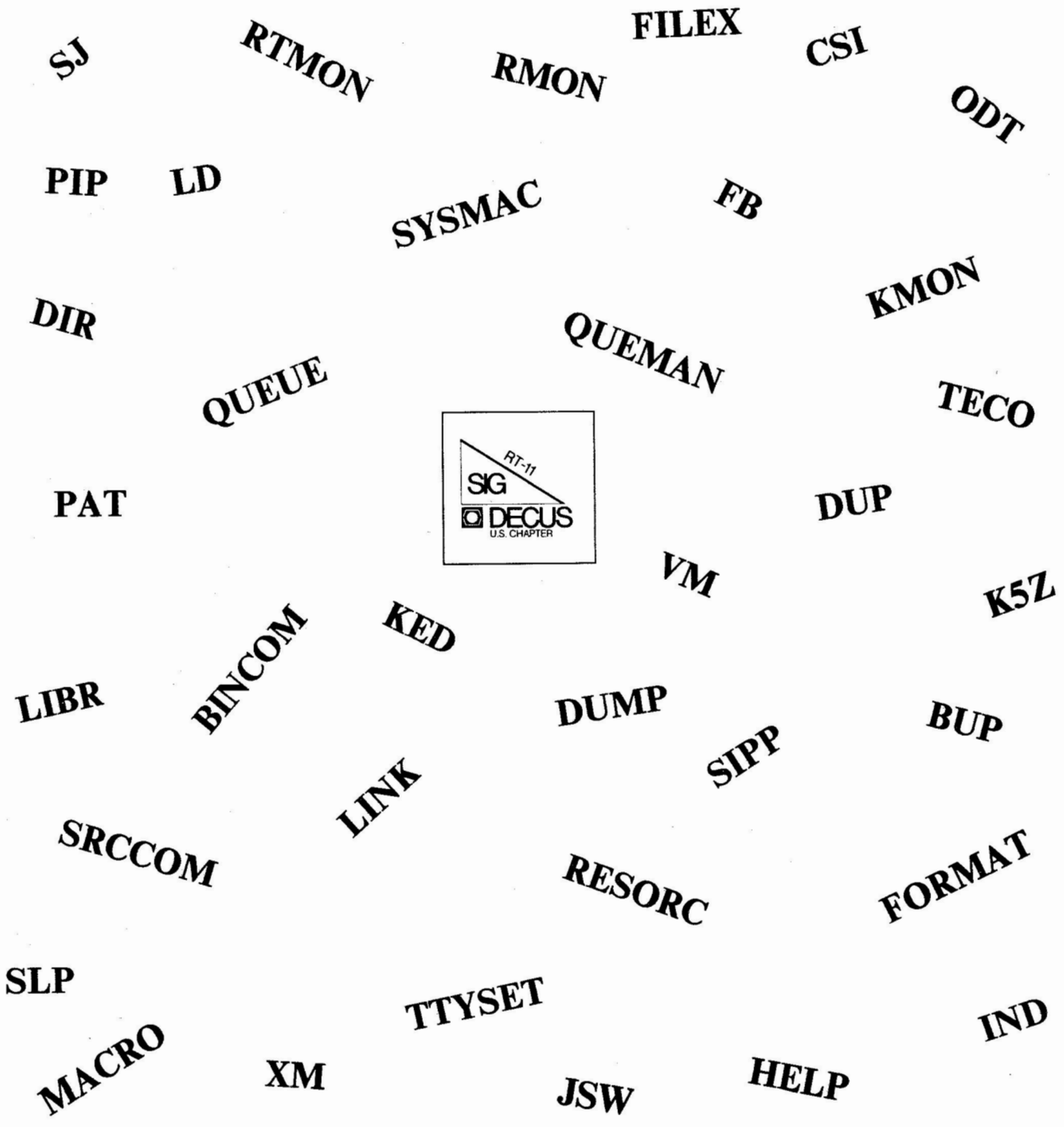
THE mini·tasker

DECUS
RT-11 SIG NEWSLETTER

October 1983 Volume 9, Number 4



Printed in the U.S.A.

THE mini• tasker

DECUS
RT-11 SIG NEWSLETTER

October 1983 Volume 9, Number 4

TABLE OF CONTENTS

THE mini• tasker

DECUS
RT-11 SIG NEWSLETTER

October 1983 Volume 9, Number 4

Contributions to the newsletter should be sent to:

Ken Demers
MS-48
United Technologies Research Center
East Hartford, Ct. 06108
(203) 727-7139 or 7240

Other communications can be sent to:

John T. Rasted
JTR Associates
58 Rasted Lane
Meriden, Ct. 06450
(203) 634-1632

RT-11 SIG
C/O DECUS
One Iron Way
MR2-3/E55
Marlboro, Ma. 01752
(617) 467-4141

From The Editor

I still need more volunteers to convert the audio tapes recorded at DECUS Symposium RT-11 sessions into articles for the "Minitasker". You will only be responsible for converting the tape from any one session. Please contact me as soon as possible.

Thankyou,

Ken Demers

USER INPUT

- DSIR
DEPARTMENT OF SCIENTIFIC AND INDUSTRIAL RESEARCH

APPLIED MATHEMATICS DIVISION

P.O. Box 1335 Wellington New Zealand
Telephone (4) 727 855 Telex 3276 Research
7th Floor Rankine Brown Building Victoria University of Wellington

SCIN 123:1 3 MAY 1983

A DOCUMENTATION DIRECTORY FOR RT-11 DEVICE HANDLERS AND INTERRUPTS.

R. D. BROWNRIGG

ABSTRACT: An exhaustive list is presented of those references to device handlers and interrupt processing contained in the DEC documentation available for the RT-11 version 4.0 operating system.

1. INTRODUCTION.

The following references are to section numbers, figures, and tables contained in the various manuals available for the RT-11 version 4.0 operating system. All references have direct relevance to device handlers in particular or interrupt processing in general. In some cases, further information is provided in parentheses to clarify exactly which aspects of these topics are mentioned or discussed in the particular section referred to. Page numbers are also provided.

The manuals referred to and their abbreviations are as follows:

MAM - DIGITAL 'microcomputers and memories' handbook (1982 edition)
GEN - RT-11 Installation and System Generation Guide
SYS - RT-11 System User's Guide
REF - RT-11 Programmer's Reference Manual
SUP - RT-11 Software Support Manual

2. DIGITAL 'microcomputers and memories' handbook

3. RT-11 Installation and System Generation Guide

4. RT-11 System User's Guide

5. RT-11 Programmer's Reference Manual

6. RT-11 Software Support Manual

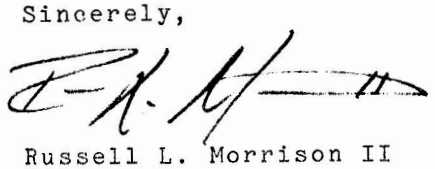
Russell L. Morrison II
Plessey Peripheral Systems
P.O. Box 19616
Irvine, CA 92714

(714)540-9945

I have discovered a small problem with RT-11 V5 and RX03 dual sided, dual density floppies: the new version has all the code that used to support RX03's deleted. What used to be a relatively simple patch to enable RX03 support is now a rather involved patch which re-enters all the old RX03 support code.

If you would like a copy of this patch please request in writing to the editor of the "Mini-Tasker".

Sincerely,



Russell L. Morrison II
Systems Analyst,
Software Support

****

HIGH LEVEL MULTILANGUAGE MACHINE-INDEPENDENT PROGRAMMATION (16, 32, 36, ... BITS) : A SUBROUTINE FOR BIT MANIPULATIONS IN BASIC AND FORTRAN IV.

****

BY DANIEL GUINIER

LABORATOIRE DE PHYSIOLOGIE COMPAREE DES REGULATIONS
GROUPE DE LABORATOIRES DU CNRS DE STRASBOURG-CRONENBOURG
23 RUE DU LOESS
B.P. 20 CR
67037 STRASBOURG CEDEX, FRANCE

INTRODUCTION :

MANIPULATION OR EXAMINATION OF BITS OF A MEMORY WORD PERMITS COMPRESSION OF BINARY DATA THAT CAN REACH A VERY INTERESTING RATE FOR STORAGE, CODING OR DATA ACQUISITION. THIS ALSO ALLOWS LOGICAL OPERATIONS APART FROM USING MACHINE CODE OR ASSEMBLER LANGUAGE WHICH ARE PARTICULAR TO A GIVEN COMPUTER.

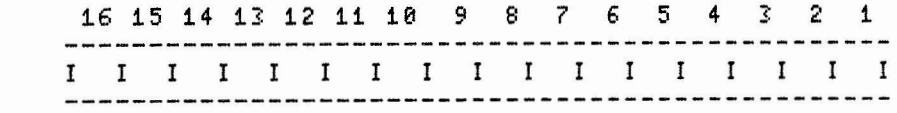
WE HAVE REALIZED A SUBROUTINE AND ITS CALLING PROGRAM WRITTEN AS IN FORTRAN IV AND ALSO IN BASIC (TESTED WITH ZX81 SINCLAIR WHICH IS THE LEAST EXPENSIVE MODEL IN THE MARKET OF MICRO-COMPUTERS). OUR PURPOSE IS TO USE THIS METHOD ON ALL TYPES AND ORGANIZATIONS OF COMPUTERS (16, 32, 36, ... BITS) AS WELL AS TO COMPARE THESE TWO LANGUAGES.

METHODS :

REPRESENTATION OF INTEGERS :

AN INTEGER I IS STORED IN A MEMORY WORD OF 16, 32, 36, ... BITS WHOSE THE HIGHEST WEIGHTED BIT IS THE BIT OF SIGN; IF THIS BIT IS RESET TO ZERO, THE NUMBER IS POSITIVE, OTHERWISE, IT IS POSITIVE.

EXAMPLE :



- LET NBRBIT=16, 32 OR 36, THE NUMBER OF BITS PER WORD.
- LET J, THE INDEX FOR THE POSITION OF THE BITS IN THE WORD (J=1 TO NBRBIT).
- LET BIT(J)=0 OR 1, THE ACTUAL VALUE OF THE J TH. BIT IN THE WORD.

IF NBRBIT=16 -(2**15-1) <= I <= 2**15-1 THAT IS -32767 <= I <= 32767
IF NBRBIT=32 -(2**31-1) <= I <= 2**31-1
IF NBRBIT=36 -(2**35-1) <= I <= 2**35-1

BINARY - DECIMAL CONVERSION :

THE VALUE OF A MACHINE WORD WHICH IS THE IMAGE OF A FIELD INTEGER I CAN BE EXPRESSED AS :

I= BIT(1) + BIT(2)*2 + BIT(3)*2**2 + BIT(4)*2**3 + BIT(NBRBIT-1)*2**(NBRBIT-2)

THE NBRBIT-TH BIT GIVES THE SIGN OF I

EXAMPLE :

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
I	I	0I	0I	0I	0I	0I	0I	0I	0I	1I	0I	0I	0I	0I	1I	1I

IF WE APPLY WHAT WAS DESCRIBED ABOVE :

I= 1 + 1*2 + 0*2**2 + ... + 1*2**6 + ... = 1 + 2 + 2**6 = 1 + 2 + 64 = 67

I = + 67 BECAUSE BIT(NBRBIT) = BIT(16) = 0

THIS IS A BINARY - DECIMAL CONVERSION.

DECIMAL - BINARY CONVERSION :

FOR A POSITIVE INTEGER I, THAT IS AN I WITHOUT ITS SIGN BIT, CAN BE CONVERTED INTO BINARY REPRESENTATION CONTAINED IN THE ELEMENTS BIT(J) OF AN ARRAY BIT( ) THAT ARE THE RESIDUALS OF SUCCESSIVE DIVISIONS PER TWO.

EXAMPLE : - TAKE THE ABSOLUE VALUE OF I = 67

67 / 2 = 33 ; RESIDUAL : BIT(1) = 1
33 / 2 = 16 ; RESIDUAL : BIT(2) = 1
16 / 2 = 8 ; RESIDUAL : BIT(3) = 0
 8 / 2 = 4 ; RESIDUAL : BIT(4) = 0
 4 / 2 = 2 ; RESIDUAL : BIT(5) = 0
 2 / 2 = 1 ; RESIDUAL : BIT(6) = 0
 1 / 2 = 0 ; RESIDUAL : BIT(7) = 1

ALL OTHER BITS FROM BIT(8) TO BIT(NBRBIT-1), THAT IS BIT(15) ARE RESET TO ZERO AND BIT(NBRBIT) = 0 IF I IS POSITIVE.

THIS IS A DECIMAL - BINARY CONVERSION.

FOR NEGATIVE NUMBERS :

IN THE PRECEDING CONVERSIONS, WE WORKED ON POSITIVE INTEGERS TO AVOID TWO'S COMPLEMENTATION, FORM IN WHICH NEGATIVE INTEGERS ARE USUALY STORED, THIS OPERATION WAS AUTOMATICALLY DONE BY A SINGLE INSTRUCTION OF SIGN CHANGE AND COMPLEMENTATION BY I = I (+/-) 2**(NBRBIT-1) WHEN BIT(NBRBIT) = 1

EXAMPLE :

	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
I = + 67	I	0I	0I	0I	0I	0I	0I	0I	0I	0I	1I	0I	0I	0I	0I	1I	1I
I = - 67	I	1I	1I	1I	1I	1I	1I	1I	1I	1I	0I	1I	1I	1I	1I	0I	1I

NUMERICAL EXAMPLES :
****

1ST. EXAMPLE : **********
****

INPUT :
I = 67
I1= 0
OUTPUT :
67
0000000001000011

2ND. EXAMPLE : **********
****
INPUT :
I =-67
I1= 0
OUTPUT :
-67
1111111110111101

3TH. EXAMPLE : **********
****
INPUT :
I = 67
I1= 1
BIT 1 = 1 BIT 9 = 1
BIT 2 = 0 BIT10 = 1
BIT 3 = 1 BIT11 = 1
BIT 4 = 1 BIT12 = 1
BIT 5 = 1 BIT13 = 1
BIT 6 = 1 BIT14 = 1
BIT 7 = 0 BIT15 = 1
BIT 8 = 1 BIT16 = 1
OUTPUT :
-67
1111111110111101

4TH. EXAMPLE : **********
****

INPUT :
I = 67
I1= 1

BIT 1 = 2	BIT 9 = 1
BIT 2 = 2	BIT10 = 1
BIT 3 = 2	BIT11 = 1
BIT 4 = 2	BIT12 = 1
BIT 5 = 2	BIT13 = 1
BIT 6 = 2	BIT14 = 1
BIT 7 = 2	BIT15 = 1
BIT 8 = 2	BIT16 = 1

OUTPUT :
-189
1111111101000011

****

LISTINGS :
****

SUBROUTINE BIT01 INCLUDES THREE PRINCIPAL PHASES :
SEARCH OF THE BITS' LEVEL FOR THE FIELD INTEGER I.
POSSIBLE CHANGES OF THIS LEVEL (0 OR 1).
RELEASING OF A NEW FIELD INTEGER I AFTER A CHANGE OF LEVEL OF THE BITS.

THESE PHASES ARE IMPLICIT FOR THE TWO VERSIONS (BASIC AND FORTRAN).
THE LISTINGS BELOW INCLUDE THE TWO VERSIONS WITH MAINS AND SUBROUTINES AND ALSO FOUR NUMERICAL EXAMPLES.

INPUT AND OUTPUT ARGUMENTS :

FORTRAN IV	BASIC	TYPE	FUNCTION
I	I	IN/OUT	INTEGER FOR THE LEVEL OF THE BITS OF THE ARRAY BIT( ), UNCHANGED FOR OUTPUT IF IND OR I1=0
IND	I1	IN	IF IND OR I1=0 : EXAMINATION OF THE BITS' LEVEL, OTHERWISE POSSIBLE CHANGE OF THESE LEVELS.
BIT( )	A( )	OUT	INTEGER ARRAY [0,1], LOADED WITH THE BITS LEVEL OF THE INTEGER FIELD I.
BITSET( )	B( )	IN	INTEGER ARRAY, GIVING THE BITS' LEVEL TO CHANGE FOR ALL BITSET( ) INCLUDED IN [0,1]
NBRBIT	N0	IN	NUMBER OF BITS IN A MACHINE WORD (NBRBIT=16, 32, 36, ... BITS).

```
      ************************************
      * LISTING OF THE FORTRAN IV VERSION *
      ************************************


      ****************************
      * MAIN PROGRAM FORTRAN IV *
      ****************************

      INTEGER BIT(36),BITSET(36)


      DATA NBRBIT/16/LEC,IMP/5,7/

      DO 1 J=1,NBRBIT
1     BITSET(J)=-1

      WRITE(IMP,2)
2     FORMAT(' $I= ')
      READ(LEC,3)I
3     FORMAT(I5)
      WRITE(IMP,4)
4     FORMAT(' $IND= ')
      READ(LEC,3)IND

      IF(IND.EQ.0)GO TO 7

      DO 6 J=1,NBRBIT
      WRITE(IMP,5)J
5     FORMAT(' $BIT',I2,' = ')
      READ(LEC,3)K
      IF(K.GT.1.OR.K.LT.0)GO TO 6
      BITSET(J)=K
6     CONTINUE

C     CALL SUBROUTINE BIT01
7     CALL BIT01(I,IND,BIT,BITSET,NBRBIT)

C     OUTPUT RESULTS.
      WRITE(IMP,8)I,(BIT(K),K=NBRBIT,1,-1)
8     FORMAT(/I8//,2X,16I1//)

      STOP
      END


      ****************************
      * SUBROUTINE FORTRAN IV *
      ****************************
C
C
      SUBROUTINE BIT01(I,IND,BIT,BITSET,NBRBIT)
C
C  DANIEL GUINIER (1983)  C.N.R.S.  STRASBOURG
C
C SUBROUTINE FORTRAN IV FOR EXAMINATION AND/OR CHANGE OF THE VALUE
C OF ONE OR SEVERAL BITS IN A MACHINE WORD FOR ANY TYPE OF COMPUTER
C ARCHITECTURE (16, 32, 36, ... BITS).
```

```
      INTEGER BIT(36),BITSET(36)


      NBR=NBRBIT-1
      IP=I
      BIT(NBRBIT)=0

      IF(IP GE.0)GO TO 1

      BIT(NBRBIT)=1
      IP=IP+2.**(NBRBIT-1)

1     DO 2 J=1,NBR

      BIT(J)=MOD(IP,2)
2     IP=IP/2

      IF(IND.EQ.0)RETURN

      DO 4 J=1,NBRBIT

      IF(BITSET(J).NE.0)GO TO 3
      BIT   (J)= 0
      BITSET(J)=-1
      GO TO 4

3     IF(BITSET(J).NE.1)GO TO 4
      BIT   (J)= 1
      BITSET(J)=-1
4     CONTINUE

      I=BIT(NBR)
      N=NBR-1

      DO 5 J=N,1,-1

5     I=I*2+BIT(J)

      IF(BIT(NBRBIT).EQ.1)I=I-2.**(NBRBIT-1)

      RETURN
      END


      ********************************
      * LISTING OF THE BASIC VERSION *
      ********************************


      ***********************
      * MAIN PROGRAM BASIC *
      ***********************

010   DIM A(36)
020   DIM B(36)

030   LET N0=16
```

```
040     FOR J=1 TO N0
050     LET B(J)=-1
060     NEXT J

070     PRINT " I ="
080     INPUT I
090     PRINT " I1="
100     INPUT I1

110     IF I1=0 THEN GOTO 190

120     FOR J=1 TO N0
130     PRINT " BIT";J;" ="
140     INPUT K
150     IF K>1 OR K<0 THEN GOTO 170
160     LET B(J)=K
170     NEXT J

180     REM " CALL SUBROUTINE BIT01
190     GOSUB 1000

200     PRINT " SORTIE DES RESULTATS"
210     PRINT " I = ";I
220     PRINT " ETAT DES BITS DE I"
230     FOR J=1 TO N0
240     PRINT A(J)
250     NEXT J

260     STOP

        ********************
        * SUBROUTINE BASIC *

1000    REM "..............................."
1010    REM " SUBROUTINE BIT01              "
1020    REM " ARGUMENTS : I,I1,A(),B(),N0"
1030    REM "..............................."

1040    LET N1=N0-1
1050    LET I2=I
1060    LET A(N0)=0

1070    IF I2>=0 THEN GOTO 1100

1080    LET A(N0)=1
1090    LET I2=I2+2**(N0-1)

1100    FOR J=1 TO N1

1110    LET I3=INT (I2/2)
1120    LET A(J)=I2-I3*2
1130    LET I2=I3
1140    NEXT J

1150    IF I1=0 THEN RETURN

1160    FOR J=1 TO N0

1170    IF B(J)<>0 THEN GOTO 1210
1180    LET A(J)= 0
1190    LET B(J)=-1
1200    GOTO 1240
```

```
1210     IF B(J)<>1 THEN GOTO 1240
1220     LET A(J)= 1
1230     LET B(J)=-1
1240     NEXT J

1250     LET I=A(N1)
1260     LET N2=N1-1

1270     FOR J=N2 TO 1 STEP -1

1280     LET I=I*2+A(J)
1290     NEXT J

1300     IF A(N0)<>1 THEN RETURN
1310     LET I=I-2**(N0-1)

1320     RETURN
```

CONCLUSION :
****

THE USER CAN DIRECTLY VERIFY OR HANDLE BITS' LEVEL OF A MEMORY WORD WHITOUT A SPECIFIC ASSEMBLER OR MACHINE CODE WHICH ARE PARTICULAR TO A GIVEN COMPUTER; THIS SUBROUTINE IS COMPLETELY TRANSPORTABLE TO ANY TYPE OF MACHINE.

THE READER WILL NOTICE SOME DIFFERENCES BETWEEN THE TWO HIGH LEVEL LANGUAGES AND ESPECIALLY WILL APPRECIATE THE MNEMONIC AND RELATIVE STATEMENTS QUALITIES OF FORTRAN.

I recently came across a problem with the RT-11 V5 Extended Memory monitor on 18-bit Qbus systems. I would like to share this problem (and its solution) with you and the other users out there.

The problem, simply (??) stated, is that DEC has been supplying the LSI 11/23+ chip set for some time, and thus some LSI systems (ours, for example) already have 22-bit addressing capability, even though the CPU is plugged into an 18-bit Qbus. Under these conditions, the memory sizing routine in RT-11 V5 does a "wrap-around", that is, the upper four bits of a 22-bit address are ignored, making the next address after "777777" equal to "000000" instead of "1000000". This is not especially critical in the Single Job monitor, where it simply causes the RESORC routine to report 4Mb of memory, and the VM Virtual Memory Disk Emulator thinks the same thing. Using the VM driver under these conditions can cause RMON to be written over and will generally cause the system to crash.

In the RT-11 V5 Extended Memory monitor, the boostrap routine sizes memory and reports to RT-11 that it has 4Mb available. This causes the XM monitor to crash on loading, so users can't even have the use of the background partition.

Since Plessey Peripheral Systems' main product line is Qbus systems, this state of affairs was wholly unacceptable, and some sort of "work around" or patch had to be developed. The result of our work consists of two unsupported patches, either or both of which may be installed to fix this problem. Since these patches alter the RT-11 source files, care must be exercised in using them; i.e., be sure you have adequate backups of your distribution.

These patches are presented as a courtesy only. We have tested these patches on the DEC RT-11 distribution and have found them to work as described. However, **Plessey Peripheral Systems makes no guarantee as to the accuracy or functionality of these patches, and will in no case provide support for systems on which they are applied. Plessey will assume no responsibility for any damages resulting from the use of these patches.**

The first patch is an addition to SYSGEN of a new parameter, **MODE22.** This parameter turns on/off 22-bit addressing in the RT-11 system, both in the extended memory monitor and the VM driver. The patch consists of three SLP files, to be applied to **SYSGEN.COM**, **BSTRAP.MAC**, and **VM.MAC**, respectively.

The first SLP file, **SYSGEN.SLP**, is as follows:

```
-/      .IFF <ESCAPE> .GOTO MO20/,.
        .IFF <ESCAPE> .GOTO Q3A
-/      .GOTO Q3/

.Q3A:   .IFF XM .GOTO MO20
        .ASK [<TRUE>] MODE22 Do you want 22-bit support (Y)?
-/      .IF DNM = "LS" .GOSUB LSC/
        .IF DNM = "VM" .GOSUB VM22
-/.CTLP90:.RETURN/

.VM22:  .IFT XM .RETURN
        ;
        .ASK [<TRUE>] MODE22 Do you want 22-bit support (Y)?
        .RETURN
-/.G5:/
        .SETS ARG "MODE22,22-bit support"
        .GOSUB SET
/
```

The second SLP file, **BSTRAP.SLP**, is as follows:

```
-/.SBTTL *        Extended Memory Bootstrap        */
.IIF NDF MODE22   MODE22   = 000000
.IIF NE MODE22    MODE22   = 000020
-/      BIS       #20,@#SR3/,/      BIT      #20,@#SR3/
.IIF NDF MODE22   MODE22   = 000000
        BIS       #MODE22,@#SR3
.IIF EQ PDT$OP    NOP
        BCS       20$
        BIT       #MODE22,@#SR3
/
```

It should be noted that the first line of this file should be read as "minus slash period SBTTL tab asterisk eight spaces Extended Memory Bootstrap nine spaces asterisk slash"

The third SLP file, **VM.SLP**, is as follows:

```
-/MODE22 = 000020/,.
.IIF NDF MODE22   MODE22   = 000000
.IIF NE MODE22    MODE22   = 000020
/
```

Once these files have been created, they may be implemented by the following commands:

```
.R SLP
*SYSGEN.COM=SYSGEN.COM,SYSGEN.SLP
*BSTRAP.MAC=BSTRAP.MAC,BSTRAP.SLP
*VM.MAC=VM.MAC,VM.SLP
*^C
```

Having implemented these files, perform a SYSGEN, or edit your **SYSGEN.CND** file to include a line:

```
MODE22    = 000000 ;22-bit support
```

which will disable 22-bit support in both the Extended Memory monitor and in the VM driver. When you perform a SYSGEN, you will notice a new question:

Do you want 22-bit support (Y)?

This question will be asked if you select the XM monitor, or, if you don't select the XM monitor, when you select the VM Virtual Memory Driver. Please note that the default base address of the VM driver in XM systems (BASE=10000) will make it uninstallable on 18-bit systems.

The second patch consists of the addition of a SET command to the VM driver. The patch is implemented through an SLP file, **VMSET.SLP**, the text of which is as follows:

```
-/REINST:/,/.EVEN/
REINST:  .ASCIZ   "?VM-W-Revome/install VM"
         .EVEN
         .IF EQ MMG$T
BAREA:   .BYTE    17,10
         .BLKW
         .BLKW
         .WORD    256.
         .WORD    0
C.BT22:  MOV      R3,V.BIT
         MOV      R3,I.BIT
         BR       PRI
         .ENDC
-/.DRSET/,.
         .IF EQ MMG$T
         .DRSET   22BIT,1,S.BT22,NO
         .ENDC
         .DRSET   BASE,1600,S.BASE,OCT
         .IF EQ MMG$T
S.BT22:  MOV      (PC)+,R3
         .WORD    21

          DEC      R3
          BR       C.BT22
          .ENDC
          .ENABL   LSB
 -/.ENDC/,/$$.SET/
          .ENDC
 PRI:     MOV      PC,R0
          ADD      #REINST-.,R0
          .PRINT
```

```
5$:
10$:    RTS     PC
        .DSABL  LSB
        $$.SET  = .
-/BIS   #MODE22,@#MMSR3/,/BEQ      20$/
        MOV     (PC)+,R1
I.BIT:  .WORD   MODE22
        MOV     #MMSR3,R2
        BIS     R1,@R2
        BCS     20$
        BIT     R1,@R2
        BEQ     20$
-/CLR   @#MMSR3/,/JMP     100$/
        CLR     @R2
        BR      100$
-/.WORD MODE22/,.
V.BIT:  .WORD   MODE22
/
```

This patch is implemented through the following commands:

```
.R SLP
*VM.MAC=VM.MAC,VMSET.SLP
*^C
```

After entering these commands, either recompile the VM driver or perform a SYSGEN. Upon completion, a command of the form:

SET VM [NO]22BIT

will be available.

Please note that these two patches are in no way incompatible; that is, they may both be in place at once. Note as well that the second patch, which implements a SET command for the VM driver only, does nothing for the Extended Memory Monitor problem. It should also be noted that, while the second patch in no way alters the functionality of the VM driver, it does change the warning message printed after a SET command from:

?VM-W-Remove and reinstall this handler

to:

?VM-W-Remove/install VM

While this is not a big deal for most users, this message might be confusing to less sophisticated users, since it can't be found in any of the manuals.

Please note that neither of these patches will correct the RESORC report that there are 4Mb of memory installed. It will, however, fix any problems connected with using the VM driver or the Extended Memory monitor. Please note as well that any software that directly manipulates the memory management registers of the 11/23 will need to be written to take the problems of the 18-bit bus into account.

I hope that these patches will be of use to those DEC and DEC compatible users who have been a little perplexed at some of the glitches in RT-11 V5's extended memory features.

Russell L. Morrison II
Plessey Peripheral Systems
P.O. Box 19616
Irvine, CA 92714

(714)540-9945

Sincerely,

The Cursor Driven Command File Driver does the following:

1. You give the program the name of a menu file which is displayed on the screen. A menu file has the name of previously created command files together with a short description.

2. You move the cursor anywhere within the command file name. This name must be alphanumeric and can be 1 to 6 characters.

3. You hit the return key. If the command file is created properly and is spelled correctly on the menu, it will then execute.

A couple of comments/observations on the program:

1. It runs on a LSI 11/23 under TSX+; a VT100 terminal in ANSI mode.

2. The escape sequences which we use as a standard at the top of EVERY menu are as follows; (I tried the program an out of date..kind of off the wall sequence and it didn't work..so be warned):

ESC = the escape char.
cr = Car. return; lf = line feed.

ESC<ESC[?31ESC[2JESC1;24rESC[HESC[0mcrlf

ESC< = If in VT52 mode reset to ANSI

ESC[?31 = If screen 132 col set to 80; the question mark is not part of the sequence but since we have C-ITOH's that use it as part of their sequence we put it in and it works on a VT100 ok.

ESC[2J = Erase entire screen.

ESC[1;24r = Set top-bottom scrolling region.

ESC[H = Cursor unconditionally to HOME position.

ESC[0m = Clear all attributes.

3. You may move the cursor anywhere with the six character command file name and the program will work. If you put a call to this program and the menu names in all command files referenced by your menu system, it can make getting around the system, significantly faster.

I'd like to thank Bruce Johnson of ITI for showing me much faster/easier ways out of the trenches at various and sundry times.

Hope you have fun with it! Any comments, experiences in implementing it, etc. would be most welcome.

I created it out of sheer frustration (ain't it always the way?), after seeing DEC350.

RONALD ROSENTHAL
HQ CECOM
U.S. ARMY ELECTRONICS COMMAND
DRSEL-MS-O
FT. MONMOUTH, NJ 07703
(201) 544-2109

```
(*$L+, $A+*)
PROGRAM PAS;
/*<!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
                                                                          ^
                                                                          \
PROGRAM LOGIC;                                                            \
        READ MENU INTO ARRAY                                              \
        MOVE CURSOR TIL CR                                                \
        GET CURSOR POSITION                                               \
        STUFF INTO X,Y                                                    \
        GET COMMAND STRING                                                \
        SETUP ASSEMBLER LINK TO COMMAND STRING INTERPRETER                \
        EXECUTE COMMAND STRING AND                                        \
        EXIT PROGRAM                                                      V
<!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>*/
  CONST
    MAXCOL=115;
    MAXROW=26;
        MINCOL=1;
        BLANK=040B;
        ESCAPE=33B;
        LEFTBR=133B;
        SIX=066B;
        SMALLN=156B;
         UPARROW=136B;
         SURPRISE=041B;
          CURSORRIGHT=103B;
          CURSORLEFT=104B;
          CURSORUP=101B;
          CURSORDOWN=102B;
                CR = 15B;
                LF = 12B;
        NINE=9;
  TYPE
    R = 1..MAXROW;
    C = 1..MAXCOL;
    MN = ARRAY [R,C] OF CHAR;
    NAME =  ARRAY [C] OF CHAR;
    CURPOS = FILE OF NAME;
        CHARSET=SET OF CHAR;
  VAR
        GOOD:BOOLEAN;
        NAMEMENU:ARRAY [1..14] OF CHAR; [MENU FILE NAME FROM COMMAND FILE]
        F: FILE OF CHAR; [ MENU FILE POINTER ]
        D:CHAR; [ RECEIVES CHARS FROM F]
        MENU:MN; [ STORES SCREENMENU IN 2D ARRAY.]
        BUFFER:NAME; [ TEMP STORAGE FOR NUMBERS FROM <READ CURSOR POSITION> ]
```

```
        ENDOFESCAPESEQUENCE: INTEGER;
        SCREEN: TEXT;
        ROW: R;
        MENUCOL, SCREENCOL, COL: C;
        INCREMENT, LEFTCOL, RIGHTCOL: INTEGER;
        LENGTHOFARRAY, LENGTHOFSTRING, DONE, COLBOUNDARY, DPOS: INTEGER;
        STOPCHAR, S, SAMPLE: CHAR;
        JOBSTAT ORIGIN 44B: INTEGER;
        CMDLENGTH ORIGIN 510B: INTEGER;
        CMDFLE ORIGIN 512B: ARRAY [1..9] OF CHAR;


/*<-------------------------------------------------------------------------->*/
PROCEDURE P020BEEP;
        CONST
                DING = 7B;
                DONG = 7B;

BEGIN
        WRITE (CHR(DING), CHR(DONG));
END; [P020BEEP]


/*<-------------------------------------------------------------------------->*/
PROCEDURE P030EXECUTECOMMANDFILE;
/* THIS POROCEDURE CALLS AN ASSEMBLY LANGUAGE MACRO CALL*/
/* TO EXECUTE THE STRING OF CHARACTERS ALREADY BUILT.    */
BEGIN
/* JOBSTAT SETS A BIT IN THE JSW INDICATING THERE'S A COMMAND FILE*/
/* TO BE EXECUTED WHEN THE EXIT MACRO IS EXECUTED.                 */

        JOBSTAT := JOBSTAT + 4000B;

[*$C
.MCALL .EXIT
CLR R0
.EXIT
*]
END; [END PROCEDURE]
/*<-------------------------------------------------------------------------->*/
PROCEDURE P040CREATECOMMANDFILE;
/*VAR INTERNAL*/
VAR CMDCOL, I1: INTEGER;
BEGIN

/* LENGTH OF COMMAND STRING SHOULD BRE SET HERE; IT WILL BE PASSED TO THE
 JOBSTATUS AREA WHEN COMMAND FILE IS EXECUTED*/

        CMDLENGTH := NINE;

                FOR I1 := 1 TO 9 DO
                        BEGIN
                        CMDFLE [I1] := ' ';
                        END;

        CMDCOL := 2;
        CMDFLE [1] := ' ';
        CMDFLE [8] := CHR(CR);
        CMDFLE [9] := CHR(LF);
```

```
                    FOR I1 := LEFTCOL TO RIGHTCOL DO
                            BEGIN
                            CMDFLE [CMDCOL] := MENU [ROW, I1];
                            CMDCOL := CMDCOL + 1;
                            END;
    END; [END PROCEDURE]


    /*<-------------------------------------------------------------------------->*/
    PROCEDURE P050TRUEFALSE (VAR CH:CHAR; SKIPSET:CHARSET; VAR DB:BOOLEAN);
    /*VAR INTERNAL*/
    BEGIN

        IF CH IN SKIPSET THEN
            BEGIN
                DB:=TRUE;
            END;
    END; [END OF P050TRUEFALSE]


    /*<-------------------------------------------------------------------------->*/
    PROCEDURE P060TESTCHAR;
    /*VAR INTERNAL*/
BEGIN
    GOOD:=FALSE;
    IF (SAMPLE >= 'A') AND (SAMPLE <= 'Z') THEN
        BEGIN
            GOOD:=TRUE;
        END;
    P050TRUEFALSE (SAMPLE,['A'..'Z'],GOOD);
    P050TRUEFALSE (SAMPLE,['0'..'9'],GOOD);
END; [END PROCEDURE]


/*<-------------------------------------------------------------------------->*/
PROCEDURE P070FINDMENUCOL (DMENU:MN;DROW:R;SCREENCOL:C;VAR ACTUALMENUCOL:C; VAR ENDE
                                                        ↳SCAPE:INTEGER);
/* THIS ROUTINE IS TO TAKE THE COLUMN NUMBER RETURNED BY THE CURSOR POSITION*/
/*  REPORT AND CORRELATE IT TO THE ACTUAL COLUMN POSITION IN THE ARRAY OF THE MENU K
                                                        ↳EPT IN CORE.*/
/*THE CPR DID NOT COUNT ESCAPE SEQUENCES AND WHEN A TAB WAS ENCOUNTERED IT*/
/* ACTUALLY INSERTED TABCOUNT (USUALLY EIGHT) NUMBER OF SPACES IN THE COLUMN*/
/* NUMBER; WHEREAS IN THE CORE ARRAY THERE IS ONLY 1 CHARACTER (11B ELEVEIN*/
/* OCTAL). */

CONST TAB=11B;TABCNT=8;

VAR DONE,APPARENTMENUCOL:INTEGER;CH:CHAR;

BEGIN
    DONE:=0;
    ACTUALMENUCOL:=0;
    APPARENTMENUCOL:=0;

REPEAT
    BEGIN

    ACTUALMENUCOL := ACTUALMENUCOL + 1;
    CH := DMENU [DROW,ACTUALMENUCOL];
IF CH = CHR(ESCAPE) THEN
```

```
                BEGIN
                                ACTUALMENUCOL := ACTUALMENUCOL + 1;
                                CH := DMENU [DROW, ACTUALMENUCOL];

                        IF CH = CHR(LEFTBR)
                        THEN
                                BEGIN
                                ENDESCAPE := ACTUALMENUCOL + 2;
                                ACTUALMENUCOL := ACTUALMENUCOL + 2;
                                END
                        ELSE

                                BEGIN
                                ENDESCAPE := ACTUALMENUCOL + 1;
                                ACTUALMENUCOL := ACTUALMENUCOL + 1;
                                END;

                                ACTUALMENUCOL := ACTUALMENUCOL + 1;
                END;

/*      THIS VARIABLE IS TO STOP THE LEFT SCAN OF P110FINDSTRINGBOUNDARY FROM*/
/* OVERSHOOTING ITS TARGET WHEN SCANNING LEFT; (IT IS ASSUMED THAT ONLY THE*/
/* HIGHLIGHT SEQUENCE <ESC[1M> STOP SEQUENCE <ESC[1M>; OR DOUBLE HEIGHT DOUBLE*/
/* WIDTH <ESC#N> WILL BE USED.) */

/*THIS COULD HAPPEN IF COMMAND FILE NAME IS RIGHT AGAINST*/

/* THE ESCAPE SEQUENCE FOR HIGHLIGHTING ON THE MENU; E.G., ESC[1MXXXXXX
N'EST PAS?? */

        IF CH = CHR(TAB) THEN
                BEGIN
                        ACTUALMENUCOL := ACTUALMENUCOL + 1;
                        APPARENTMENUCOL := APPARENTMENUCOL + TABCNT;
                END;

        IF  CH = CHR(CR) THEN
                BEGIN
                        DONE:=1;
                END;


        IF CH > CHR(37B) THEN
                BEGIN
                        APPARENTMENUCOL := APPARENTMENUCOL + 1;
                END;
        IF APPARENTMENUCOL = SCREENCOL  THEN
                BEGIN
                        DONE:= 1;
                END;

        END
UNTIL DONE = 1;


END; [P070FINDMENUCOL]
```

```
/*<-------------------------------------------------------------------------->*/
PROCEDURE P080TESTFORESCAPEBOUNDARY (DCOL,ENDCOL:INTEGER; VAR DB:BOOLEAN);

BEGIN
        IF DCOL = ENDCOL        THEN
                BEGIN
                DB := FALSE;
                END;

END; [P080TESTFORESCAPEBOUNDARY]


/*<-------------------------------------------------------------------------->*/
PROCEDURE P090TESTFORSTRINGLENGTH (COLNOW,COLBEGIN,LS:INTEGER; VAR DB:BOOLEAN);
/*VAR INTERNAL*/
BEGIN
        IF ABS (COLNOW - COLBEGIN) > LS THEN
                BEGIN
                DB := FALSE;
                END;
END; [END PROCEDURE]


/*<-------------------------------------------------------------------------->*/
PROCEDURE P100TESTCOLIMITS (DCOL,MIN,MAX :INTEGER; VAR DB:BOOLEAN);
/*VAR INTERNAL*/
BEGIN
        IF (DCOL <  MIN) OR (DCOL >  MAX)  THEN
                BEGIN
                DB := FALSE;
                END;

END; [END PROCEDURE]
/*<-------------------------------------------------------------------------->*/
PROCEDURE P110FINDSTRINGBOUNDARY (ENDESCAPE:INTEGER; DMENU:MN; DROW:R; DCOL:C;
                                        ↳ VAR DCOLBOUNDARY:C; VAR  INC:INTEGER);
/*VAR INTERNAL*/
VAR NUMCHARS,DLENGTHOFSTRING :INTEGER;

BEGIN

NUMCHARS := 0;
DLENGTHOFSTRING := 6;
COLBOUNDARY:=DCOL;
DONE := 0;

REPEAT
        GOOD := TRUE;
        SAMPLE := DMENU [DROW,COLBOUNDARY];
        P060TESTCHAR;

        IF GOOD = TRUE THEN
                BEGIN
                P080TESTFORESCAPEBOUNDARY (COLBOUNDARY,ENDESCAPE,GOOD);
                END;

        IF GOOD = TRUE THEN
                BEGIN
                P090TESTFORSTRINGLENGTH (COLBOUNDARY,DCOL,DLENGTHOFSTRING,GOOD);
                END;
```

```
        IF GOOD = TRUE THEN
                BEGIN
                P100TESTCOLIMITS (COLBOUNDARY, MINCOL, MAXCOL, GOOD);
                END;

        IF GOOD = TRUE THEN
                                BEGIN
                                        COLBOUNDARY:= COLBOUNDARY + INC;
                                        NUMCHARS:=NUMCHARS + 1;
                                END
                        ELSE
                                BEGIN
                                        DONE := 1;
                                        COLBOUNDARY := COLBOUNDARY - INC;
                                END;
UNTIL DONE = 1;

/* IF SOMEONE PUTS THE CURSOR ON A NO-NO WE WANT TO MOVE IT BACK TO ITS */
/* STARTING POINT. */

        IF NUMCHARS = 0 THEN
                BEGIN
                                COLBOUNDARY:= COLBOUNDARY + INC;
                END;

DCOLBOUNDARY := COLBOUNDARY;

END; [END PROCEDURE]
/*<-------------------------------------------------------------------------->*/
PROCEDURE P120GETCHAR;
/*VAR INTERNAL*/
/*THIS PROCEDURE IS FOR DIPLSAY THE MENU*/
BEGIN
                GET(F);
                D:=F^;
END; [P120GETCHAR]

/*<-------------------------------------------------------------------------->*/
 PROCEDURE P130DISPLAYSTOREFILE;
 /*VAR INTERNAL*/
 BEGIN

          READ (NAMEMENU);
          RESET (F,NAMEMENU);
 COL :=1;
 ROW :=1;

 WHILE NOT EOF(F) DO BEGIN

 /* DISPLAY MENU AND STORE IN ARRAY IN CORE;
   DISPLAY SCREEN
 */

          D:=F^;

          IF D=CHR(000B) THEN BEGIN
                             END
          ELSE
                             BEGIN
                WRITE(D);
                MENU[ROW,COL]:=D;
                COL:=COL+1;
```

```
                        END;
IF D = CHR(LF) THEN
        BEGIN
                        COL:=1;
                        ROW:=ROW+1;
        END;

/*                      ONE READ ONE WRITE PER MODULE!!!!!!*/

                                P120GETCHAR;

END; [END OF WHILE]

                        CLOSE(F);

END; [P130DISPLAYSTOREFILE]


/*<-------------------------------------------------------------------------->*/
PROCEDURE P140ECHOFF;
/*VAR INTERNAL*/
        BEGIN
                WRITE(CHR(035B),'F');
END; [P140ECHOFF]


/*<-------------------------------------------------------------------------->*/
PROCEDURE P150ECHON;
/*VAR INTERNAL*/
        BEGIN
                WRITE(CHR(035B),'E');
END; [P150ECHON]


/*<-------------------------------------------------------------------------->*/
PROCEDURE P160GETCURSORPOSITION;
                        BEGIN
                                WRITE(CHR(ESCAPE));
                                WRITE(CHR(LEFTBR));
                                WRITE(CHR(066B));
                                WRITE(CHR(156B));
END; [P160GETCURSORPOSITION]
/*<-------------------------------------------------------------------------->*/
PROCEDURE P170GETCURSORPOSITION;
BEGIN
P160GETCURSORPOSITION;
END; [P170GETCURSORPOSITION]


/*<-------------------------------------------------------------------------->*/
PROCEDURE P180GETSCREENCHAR (VAR CH:CHAR);

/* THIS PROCEDURE WILL BE MADE AN EXTERNAL PROCEDURE THAT*/
/* WILL BE CALLABLE FROM ANY PASCAL PROGRAM. */

VAR JOBSTAT ORIGIN 44B:INTEGER;

        /* B & C ARE ON AND OFF VT100 ESCAPE LETTER ACTIVATION.  */
        /* S & T ARE ON AND OFF SINGLE CHAR. ACTIVATION  */
        /* WHEN ON YOU DON'T NEED A CR TO RETURN A CHAR TO YOU */
```

```
        /* JOBSTAT DOES THE SAME THING FOR RT11 AS S DOES FOR TSX+ */
        /* SEE PROCEDURE P085.. FOR TURNING OPTIONS OFF. */
BEGIN
        WRITE (CHR(035B),'B');
        WRITE (CHR(035B),'S');
        JOBSTAT := JOBSTAT OR 10000B;

        REPEAT
                        [$C
                        .MCALL .TTYIN
                        .TTYIN
                        MOVB R0,@CH(6)
                        ]
        UNTIL CH # CHR(0);


END; [P180GETSCREENCHAR]


/*<-------------------------------------------------------------------------->*/
PROCEDURE P190REVERSEP180OPTIONS;

/* THIS PROCEDURE WILL BE MADE AN EXTERNAL PROCEDURE THAT*/
/* WILL BE CALLABLE FROM ANY PASCAL PROGRAM.*/

VAR JOBSTAT ORIGIN 44B:INTEGER;

        /* B & C ARE ON AND OFF VT100 ESCAPE LETTER ACTIVATION. */
        /* S & T ARE ON AND OFF SINGLE CHAR. ACTIVATION */
        /* WHEN ON YOU DON'T NEED A CR TO RETURN A CHAR TO YOU */

        /* JOBSTAT DOES THE SAME THING FOR RT11 AS S DOES FOR TSX+ */
        /* SEE PROCEDURE P080.. FOR TURNING OPTIONS ON. */
BEGIN
        WRITE (CHR(035B),'C');
        WRITE (CHR(035B),'T');
        JOBSTAT := JOBSTAT AND NOT 10000B;

END; [END PROCEDURE P190REVERSEP180OPTIONS]


/*<-------------------------------------------------------------------------->*/
PROCEDURE P200SCANTIL;

VAR X:INTEGER;
 BEGIN
         FOR X:= 1 TO LENGTHOFARRAY DO
                 BEGIN
                         BUFFER[X] := ' ';
                 END;

 X:=0;

                 WHILE S # STOPCHAR DO
                         BEGIN
                                 X:=X+1;
                                 BUFFER[X] := S;
                                 P180GETSCREENCHAR(S);
                         END;

                 LENGTHOFSTRING := X;
```

```
END; [P200SCANTIL]


/*<-------------------------------------------------------------------------->*/
PROCEDURE P210ARRAYTONUM;
/*VAR INTERNAL*/

VAR I1, I2: INTEGER; R1: REAL;
BEGIN
R1:=0.0;
I2 := 0;
DPOS:=0;
FOR I1:=LENGTHOFSTRING DOWNTO 1 DO
        BEGIN
                R1 := R1+((ORD(BUFFER[I1]) - ORD('0')) * EXP10(I2));
                I2 := I2 + 1;
        END;
DPOS := TRUNC(R1);
END; [P210ARRAYTONUM]


/*<-------------------------------------------------------------------------->*/
PROCEDURE P220MOVECURSOR;
VAR ENDSW: INTEGER;
BEGIN
ENDSW := 0;

REPEAT
                BEGIN

P180GETSCREENCHAR(S);

IF S = CHR(ESCAPE) THEN BEGIN
                                P180GETSCREENCHAR(S);
                            END;
IF S = CHR(LEFTBR) THEN BEGIN
                                P180GETSCREENCHAR(S);
                            END;
IF S = CHR(CURSORRIGHT) THEN    BEGIN
                                WRITE (CHR(ESCAPE));
                                WRITE (CHR(LEFTBR));
                                WRITE (CHR(103B));
                                END;
IF S = CHR(CURSORLEFT) THEN BEGIN
                                WRITE (CHR(ESCAPE));
                                WRITE (CHR(LEFTBR));
                                WRITE (CHR(CURSORLEFT));
                            END;
IF S = CHR(CURSORUP) THEN BEGIN
                                WRITE (CHR(ESCAPE));
                                WRITE (CHR(LEFTBR));
                                WRITE (CHR(CURSORUP));
                            END;
IF S = CHR(CURSORDOWN) THEN BEGIN
                                WRITE (CHR(ESCAPE));
                                WRITE (CHR(LEFTBR));
                                WRITE (CHR(CURSORDOWN));
                            END;

/*IF S = CHR(CR) THEN   BEGIN
                                ENDSW := 1;
                            END;
```

```
*/
IF S = CHR(LF) THEN      BEGIN
                                 ENDSW := 1;
                         END;

                END [END OF REPEAT]
UNTIL ENDSW = 1;
END; [P220MOVECURSOR]


/*<-------------------------------------------------------------------------->*/
PROCEDURE P230STUFFINTOXY;
/*VAR INTERNAL*/
BEGIN
STOPCHAR := ';';
LENGTHOFARRAY := 2;
P180GETSCREENCHAR(S);
P180GETSCREENCHAR(S);
P180GETSCREENCHAR(S);
P200SCANTIL;
P210ARRAYTONUM;
ROW := DPOS;
P180GETSCREENCHAR(S);
STOPCHAR := 'R';
P200SCANTIL;
P210ARRAYTONUM;
SCREENCOL := DPOS;
END; [P230STUFFINTOXY]


/*<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$>*/


/*MAIN PROGRAM*/

BEGIN
P130DISPLAYSTOREFILE;
P220MOVECURSOR;
P160GETCURSORPOSITION;
P230STUFFINTOXY;
/* THESE NEXT TWO STATEMENTS ARE FOR DEBUGGING PURPOSES COMMENTED OUT */
/*WRITE ('SCREEN', SCREENCOL); */
/*WRITE ('ROWXXX', ROW); */
P070FINDMENUCOL (MENU, ROW, SCREENCOL, MENUCOL, ENDOFESCAPESEQUENCE);
INCREMENT:=-1;
P110FINDSTRINGBOUNDARY (ENDOFESCAPESEQUENCE, MENU, ROW, MENUCOL, LEFTCOL, INCREMENT);
INCREMENT:=1;
P110FINDSTRINGBOUNDARY (ENDOFESCAPESEQUENCE, MENU, ROW, LEFTCOL, RIGHTCOL, INCREMENT);
P040CREATECOMMANDFILE;
/* SAME METHOD USED HERE FOR THIS DEBUGGING STATEMENT */
/*WRITE (LEFTCOL, RIGHTCOL, CMDFLE); */
P190REVERSEP180OPTIONS;
P030EXECUTECOMMANDFILE;
END.
```

USER REQUESTS

I am running RT-11 on a PDP 11/23 to prepare a TU-58 to run stand-alone on a 11/04. What do I have to change in FRT.MAC which is part of the stand-alone module? Where can I get the latest documentation on FRT.MAC and SIMRT.MAC.

Joseph F. Heinig
NASA Goddard Space Flight Center
Code 564.3
Advanced Systems Section
Greenbelt Road
Greenbelt, Md. 20771

We are in urgent need of a serial handler for a printer with X-on/X-off protocol for operation under Version 3 of RT-11.

If you can advise us as to where we might find such a handler, we would be most grateful.

Very truly yours,

VARTRON CORPORATION
750 WELCH ROAD
PALO ALTO, CALIFORNIA 94304
PHONE: (415) 328-2531

Pat Vartanian
DECUS Associate 118501

USER RESPONSES

The very usefull programm DATE, published by R.M.Harrington in Mini-tasker March 1983 Vol 9, No.1, could be added with some lines to accept time from 20: to 23: hours (for night-workers!)

Change the lines between the comment "HH OR H FORMAT" and "NOW CHECK FOR ERRORS" as follows:

```
;    HH   OR   H   FORMAT
;
HH:         CMP         R1,#541
            BPL         TIM                     ; too much input
            MOV         #34460, R2              ; Range 0-9
            JSR         PC, CHECKR
            CMPB        -1 (R1),#40             ; CK for Space
            BEQ         CKE
```

```
        CMPB    -1 (R1),#61         ; CK for 1
        BEQ     ADJ
        CMPB    -1 (R1),#62         ; CK for 2

        BNE     TIM
        INC     R1
        MOV     #31460, R2          ; Range 0-3
        JSR     PC, CHECKR
ADJ:    DEC     R1                  ; Adjust Pointer
        CMPB    -1 (R1),#40         ; CK for Space
        BNE     TIM
CKE:    CMPB    -2 (R1),#'E         ; CK for 'E' of time
        BNE     TIM

;
;    NOW CHECK FOR ERRORS
```

Yours sincerely

L. Kahlbau

L.Kahlbau
c/o SIEMENS,SARL
Fertigungstechnik
Estr.de Almeirim
7000 Evora
PORTUGAL

UPCOMING SYMPOSIUM INFORMATION

ANNOUNCING

RT-11 SESSION NOTES

for

DECUS LAS VEGAS

There will be a volume of Session Notes containing the visuals for some of the RT-11 papers to be given at the DECUS 1983 Fall Symposium in Las Vegas. Additionally, the volume will contain "The Best of RT-11, Volume 2", as an added bonus.

Look for the document at the DECUS store at the Symposium.

1983 FALL SYMPOSIA IN LAS VEGAS

Even though there was very little time between the Spring symposia and the Fall symposia, I received a record number of submissions for the RT-11 SIG. The scheduling problem was compounded by the fact that the number of meeting rooms was less than before. The end result of all this is a schedule that is a little different than before. First of all, we will be starting at 8:30 in the morning instead of 9:00. Second, the coffee break time was eliminated so that we could hold more sessions. Cookies and milk will be available in a number of locations so that you will not have to go the entire morning or afternoon without food. Finally, you will notice that a number of RT-11 sessions are scheduled for Friday morning and afternoon. The thought here was that this was really a five-day convention and that we should better utilize Friday in order to avoid "session burnout." As always, I will be available at the symposia to listen to constructive criticism only so long as you buy the beer.

I have attached a Master Index of all the RT-11 sessions to enable you to make plans to attend the symposia. I hope to see you there.

RT-11

CODE	TITLE/SPEAKER	TIME REQ.
R001	USING A PDP-11/23 AS A FILE SERVER FOR ATTACHED LSI-11'S Fouts, Martin	1 hour
R002	MIGRATION OF DBMS SOFTWARE FROM RT-11 TO RSX-11M Natale, Robert C.	30 minutes
R004	RT-11 USERS SPEAKOUT Rhodes, Ned W.	2½ hours
R005	RT-11/TSX-PLUS COMPATIBILITY ISSUES Peterson, Jack J.	1 hour
R007	RT-11 SIG BUSINESS MEETING Rasted, John T.	30 minutes
R008	RT-11 SIG SYMPOSIUM WRAP-UP Rasted, John T.	30 minutes
R009	DECUS LIBRARY LAYERED PRODUCTS PANEL FOR RT-11 Bourgeois, Nick	1½ hours
R010	RT-11 ROADMAP Rasted, John T.	30 minutes
R011	RT-11 USER APPLICATION WORKSHOP Rasted, John T.	1 hour
R012	RT-11 USER COMMAND LINKAGE Crowell, John M.	30 minutes

R013	RT-11 FUTURES WORKSHOP Crowell, John M.	1 hour
R014	HOW TO DEVELOP RT-11 DEVICE HANDLERS Rhodes, Ned W.	1 hour
R017	COMBATTING FLASH FLOODS WITH PDP-11S Peterson, Jack J.	1 hour
R018	TSX-PLUS INTERNALS Bramlet, Jan	1 hour
R019	ACCESSING MEMORY ABOVE 56KB FROM RT-11 FORTRAN Trellue, Ron	1 hour
R020	SHARED REGIONS AND RESIDENT LIBRARIES FOR RT-11 XM Adams, Greg	1 hour
R021	RT-11 XM NEW USER Adams, Greg	1 hour
R022	RT-11 FEEDBACK SESSION RT-11 Software Development Group	1 hour
R023	RT-11 LANGUAGES PANEL RT-11 Software Development Group	30 minutes
R024	RT-11 PRODUCT PANEL RT-11 Software Development Group	1 hour
R025	RT-11 DIRECTORY STRUCTURES INTERNALS Gentry, Martin	1 hour
R027	RT-11 IND NEW USER Metsch, James	1 hour
R029	USING TSX-PLUS SHARED RUN-TIME SYSTEMS Crapuchettes, Jim	1 hour
R030	IMPROVING PERFORMANCE OF RT-11 FORTRAN PROGRAMS Crapuchettes, Jim	1 hour
R031	TSX-PLUS REAL-TIME I/O TECHNIQUES Crapuchettes, Jim and Clark, Tim	1 hour
R032	TSX-PLUS QUESTION & ANSWER AND MAGIC Kingsbury, Dan	1 hour

PAST SYMPOSIUM INFORMATION

From: William K. Walker
Monsanto Research Corp.
P. O. Box 32 OS-123
Miamisburg, Ohio 45342
(513) 865-3557

I gave a short presentation during the Foreign Peripherals Forum at the St. Louis DECUS meeting on the model 306A clock board from Grant Technology Systems. This is a KWV11-C equivalent board that also includes a really slick battery-backed calendar clock option. A number of people expressed interest in a couple of utilities which I had written to set the calendar clock and to set the RT-11 date and time from the clock values. This stuff was not ready for the RT-11 SIG tape at the time and I didn't have any listings with me. I have since found time to clean-up these routines and to add some additional code and assembly conditionals to make them more general. I am enclosing source listings for those of you who might be interested. For those of you who are too lazy to do your own typing, I have also submitted them to the DECUS library.

Sincerely,

William K. Walker

```
.title  set306.mac
.enabl  lc
.ident  /wkw02/
.nlist  cnd

; This program sets up the date and time on the calendar clock option for
; the GTSC model 306A real-time clock/calendar clock board.

; This is NOT a real sophisticated program -- if you tell it to set up
; garbage on the board, it will cheerfully do so. It is, however, simple,
; and relatively easy to understand.

; Note that there are conditionals in the code for three different set-up
; variables. You may choose to read/write the registers in binary or BCD
; format, you may keep AM/PM or 24-hour clock time, and you can have the
; board compensate for Daylight Savings Time. Note also that the program
; turns all interrupt enable bits off. The alarm times are undisturbed
; however.

; This program will run under versions 4 and 5 of RT-11 and probably
; earlier and later versions as well.

; Contributed by:       William K. Walker
;                       Monsanto Research Corp.
;                       P. O. Box 32     OS-123
;                       Miamisburg, Ohio  45342
```

```
.mcall  .stlin, .exit

ccba    = 170400        ;Base address for clock registers
resa    = ccba+12       ;Register A address
resb    = ccba+13       ;Register B address
resd    = ccba+15       ;Register D address

; Conditionals:

dm      = 4     ;Disable if board is to operate in BCD mode
ck24    = 2     ;Disable if board is to keep AM/PM time
dse     = 1     ;Disable if board is not to compensate for daylight savings
                ; time

.iif ndf dm,    dm    = 0
.iif ndf ck24,  ck24  = 0
.iif ndf dse,   dse   = 0

set     = 200
dvrset  = 160
dvset   = 40
bset    = set!dm!ck24!dse

set306:
        mov     #ccba,r0        ;Set up to grab current time data off board
        mov     #sec,r1         ; (really just after alarm times)...
        mov     #6,r2
10$:    tstb    @#resa          ;Update in progress?
        bmi     10$             ;Branch if so
20$:    movb    (r0)+,(r1)+     ;Get the data...
        sob     r2,20$

        movb    @#resd,r0       ;Set 'valid RAM and time' bit
        .stlin  #buf,#yearq     ;Prompt for and set year
        call    ascbin          ;Convert to appropriate binary
        movb    r0,year         ;Store result
        .stlin  #buf,#monthq    ;Get, convert, and store month...
        call    ascbin
        movb    r0,month
        .stlin  #buf,#dayq      ;Do day of month...
        call    ascbin
        movb    r0,day
        .stlin  #buf,#daywkq    ;Do day of week...
        call    ascbin
        movb    r0,daywk
        .stlin  #buf,#hourq     ;Hour...
        call    ascbin
        movb    r0,hour
.if eq  ck24
        .stlin  #buf,#ampmq     ;"AM or PM?"
        bicb    #240,buf        ;Make response upper case, 7-bit
        cmpb    buf,#'A         ;AM?
        beq     30$
        cmpb    buf,#'P         ;PM?
        bne     30$
        bisb    #200,hour       ;Set PM (high-order) bit
.endc
30$:    .stlin  #buf,#minq      ;Minute...
        call    ascbin
        movb    r0,min
```

```
        .stlin  #buf,#seca      ;Second...
        call    ascbin
        movb    r0,sec

        .stlin  #buf,#seta      ;"Hit <return> to set clock..."
        mov     #sec,r0         ;R0 => data buffer
        mov     #ccba,r1        ;R1 => clock registers
        mov     #10.,r2         ;R2 = no. of bytes to transfer
        movb    #bset,@#resb    ;Toggle set-up bits
        movb    #dvrset,@#resa  ;Reset divider chain
40$:    movb    (r0)+,(r1)+     ;Load registers...
        sob     r2,40$
        movb    #dvset,@#resa   ;Remove divider reset
        bicb    #set,@#resb     ;Start clock
        .exit                   ;Exit to RT-11

ascbin:
        mov     #buf,r1         ;R1 => input buffer
        clr     r0              ;Clear R0; will contain result
        tstb    (r1)+           ;Test first character
        bne     10$             ;Continue if not null
        return                  ;Return with zero result otherwise

10$:    tstb    (r1)            ;Test 2nd character
        beq     20$             ;If null, number is 0-9
        movb    -(r1),r0        ;Get 10's character
        bic     #^C<17>,r0      ;Strip out ASCII stuff
.if eq  dm
        .rept   4               ;Move left 4 bits into high-order nibble
        asl     r0
        .endr
.iff
        asl     r0              ;Multiply by 10. by doing (n*2)+(n*8)...
        mov     r0,r2
        asl     r0
        asl     r0
        add     r2,r0
.endc
        tst     (r1)+           ;Adjust pointer

20$:    movb    -(r1),r2        ;Get one's character
        bic     #^C<17>,r2      ;Strip out ASCII stuff
.if eq  dm
        bis     r2,r0           ;Set low-order nibble
.iff
        add     r2,r0           ;Add to result for 10's character
.endc
        return

buf:    .blkb   134.    ;Buffer for .stlin request

sec:    .byte   0       ;Data buffer for clock registers...
        .byte   0
min:    .byte   0
        .byte   0
hour:   .byte   0
        .byte   0
daywk:  .byte   0
day:    .byte   0
month:  .byte   0
year:   .byte   0
```

```
; Prompt messages:

yearq:  .ascii  /     Year (1983=83)? /<200>
monthq: .ascii  /      Month (Jan=1)? /<200>
dayq:   .ascii  /                Day? /<200>
daywkq: .ascii  /Day of week (Sun=1)? /<200>
hourq:  .ascii  /               Hour? /<200>
ampmq:  .ascii  /  AM or PM (A or P)? /<200>
minq:   .ascii  /             Minute? /<200>
secq:   .ascii  /             Second? /<200>
setq:   .ascii  /Hit <return> to set clock.../<200>
        .even

.end    set306
```

```
.title  setdt.mac
.ident  /wkw02/
.enabl  lc
.nlist  cnd

; This program sets the RT-11 date and time  from  the  GTSC  model  306A
; clock board.

; The handiest way to use this program is to  run  it  in  your  start-up
; command file.  It can, of course, be run at any time you  may  wish  to
; bring the RT-11 date and time into agreement with the the clock on  the
; 306A.

; This program will run under a version 4 or 5 monitor.

; Contributed by:       William K. Walker
;                       Monsanto Research Corp.
;                       P. O. Box 32     OS-123
;                       Miamisburg, Ohio  45342

.mcall  .gval, .print, .sdttm, .exit

ccba    = 170400        ;Calendar clock base address
rega    = ccba+12       ;"Register A" address


jsw     = 44    ;Address of job status word
userrb  = 53    ;Address of user error byte
sysver  = 276   ;Offset of monitor version number
sever$  = 10    ;Severe error bit in user error byte

eis     = 1     ;Disable if you do not have EIS instructions
;lt50hz = 1     ;Remove semicolon if you have 50Hz line-time clock
;ampm   = 1     ;Remove semicolon if board is set up for AM/PM time
;bcd    = 1     ;Remove semicolon if board is set up in BCD mode
datime  = 1     ;Disable if you don't want date and time printed on exit


; NOTE: The mainline code in this program doesn't really do  very  much.
;       If you strip out this mainline  code  and  add  the  appropriate
;       GLOBL and PSECT stuff, you  have  a  MACRO  or  FORTRAN-callable
;       subroutine named GSDTM that will set the  RT-11  date  and  time
;       from the 306A clock.
```

```
setdt:
        .gval   #area,#sysver   ;Get RT-11 version number...
        bic     #^C<377>,r0
        cmpb    r0,#4           ;Test for version 4
        bgt     20$             ;Branch if V5 or later
        beq     10$             ;Branch if V4

        .print  #wrsver         ;Complain if earlier than V4
        bisb    #sever$,@#userrb ;Set severe error bit
        clr     r0              ;Do a hard exit...
        .exit

10$:    mov     #4000,chnbit    ;Set-up for V4-style chain exit

20$:    call    gsdtm           ;Set date and time from 306A
.if df  datime
        mov     #1000,sp        ;Pass DATE and TIME commands to RT-11
        mov     csize,r0        ; on exit...
        mov     #cstart,r1
        mov     #510,r2
        mov     r0,(r2)+
30$:    movb    (r1)+,(r2)+
        sob     r0,30$
        bis     chnbit,@#jsw
40$:    clr     r0
.endc
        .exit

chnbit: .word   40
wrsver: .asciz  /?SETDT-F-Wrong Version Of RT-11/
        .even
.if df  datime
csize:  .word   cend-cstart
cstart: .asciz  /DATE/
        .asciz  /TIME/
cend:   .even
.endc

; "Get-and-set" date and time from GTSC model 306A clock board.

.iif ndf bcd, noon = 12.
.iif df  bcd, noon = 22         ;(22 = 12. in BCD)

gsdtm:
        mov     #10.,r3         ;Get ready to move 10 bytes
        mov     #ccba,r4        ; starting from the cc base address
        mov     #sec,r5         ; to the local buffer
10$:    tstb    @#rega          ;Clock update in progress?
        bmi     10$             ;Branch if so
20$:    movb    (r4)+,(r5)+     ;Move a byte to the buffer
        sob     r3,20$          ;Keep going until done

.if df  ampm                    ;Code for AM/PM time...
        tstb    hr              ;Is time PM?
        beq     30$             ;Branch if not
        movb    hr,r0           ;Get hours value
        bic     #^C<17>,r0      ;Mask for hour value
        add     #noon,r0        ;Make it a 24-hour time value
        movb    r0,hr           ;Store "corrected" value
30$:
.endc
```

```
.iif df bcd, call bcdbin        ;Convert register values to binary,
                                ; if necessary

        clr     r5              ;Clear R5
        movb    month,r5        ;Get the month
.if ndf eis
        .rept   5               ;Shift left 5 bits...
        asl     r5
        .endr
.iff
        ash     #5,r5           ;Shift left 5 bits, EIS code
.endc

        bisb    daym,r5         ;Get the day of the month
.if ndf eis
        .rept   5               ;Shift left 5 bits...
        asl     r5
        .endr
.iff
        ash     #5,r5           ;Shift left 5 bits, EIS
.endc

        movb    year,r4         ;Get the year
        sub     #72.,r4         ;Offset it from 1972
        bis     r4,r5           ;Stuff it into r5
        mov     r5,date         ;Save result as date word

        clr     r0              ;Clear R0 (will be high-order time)
        clr     r1              ;Clear R1 (will be low-order time)
        movb    hr,r1           ;Get hour and
        call    muld60          ; convert it to minutes past midnight
        movb    min,r5          ;Get minutes,
        add     r5,r1           ; add them in, and
        call    muld60          ; convert to seconds
        movb    sec,r5          ;Get seconds
        add     r5,r1           ;Add to get seconds from midnight...
        adc     r0
.if ndf lt50hz
         call    muld60          ;Convert to ticks (60Hz clock)
 .iff
         call    muld50          ;Convert to ticks (50Hz clock)
 .endc
         mov     r0,timehi       ;Store high-order time
         mov     r1,timelo       ;Store low-order time

         .sdttm  #area,#date     ;Set the RT-11 date and time

         return

 ; Buffer for clock registers:

 sec:    .byte   0
         .byte   0
 min:    .byte   0
         .byte   0
 hr:     .byte   0
         .byte   0
         .byte   0
 daym:   .byte   0
 month:  .byte   0
 year:   .byte   0
```

```
; Date and time words for .SDTTM request:

date:   .word   0
timehi: .word   0
timelo: .word   0

area:   .word   0,0     ;EMT argument block for .SDTTM

; This routine multiplies a double-precision (two-word) integer by 60.
; It takes advantage of this special case, and does it as (64*N)-(4*N).

muld60:
.if ndf eis                         ;Non-EIS code...
        mov     #2,r4               ;Multiply R0,R1 by 4...
        clc
10$:    asl     r1
        rol     r0
        sob     r4,10$
        mov     r0,r2               ;Save result (n*4)...
        mov     r1,r3

        mov     #4,r4               ;Now multiply by 16 to get n*64
        clc
20$:    asl     r1
        rol     r0
        sob     r4,20$
.iff                                ;EIS code...
        ashc    #2,r0               ;Multiply R0,R1 by 4...
        mov     r0,r2               ;Save result...
        mov     r1,r3
        ashc    #4,r0               ;Now multiply by 16 to get n*64
.endc

        sub     r3,r1               ;Subtract R2,R3 from R0,R1...
        sbc     r0
        sub     r2,r0
        return

.if df  lt50hz

; For those of you with 50Hz line-time-clocks, this routine multiplies
; a two-word integer in R0,R1 by 50.  It does this by treating 50*n as
; (32+16+2)*n.

muld50:
.if ndf eis                         ;Non-EIS code...
        clc                         ;Multiply R0,R1 by 2...
        asl     r1
        rol     r0
        mov     r0,-(sp)            ;Save result (n*2)
        mov     r1,-(sp)

        mov     #3,r4               ;Now multiply by 8 to get n*16...
        clc
10$:    asl     r1
        rol     r0
        sob     r4,10$
        mov     r0,-(sp)            ;Save result (n*16)
        mov     r1,-(sp)
```

```
        clc                     ;Now multiply by 2 so that R0,R1 is n*32...
        asl     r1
        rol     r0

.iff                            ;EIS code to do same as above...
        ashc    #1,r0           ;n*2...
        mov     r0,-(sp)
        mov     r1,-(sp)

        ashc    #3,r0           ;n*16...
        mov     r0,-(sp)
        mov     r1,-(sp)

        ashc    #1,r0           ;n*32...

.endc
        add     (sp)+,r1        ;Add n*16 to n*32 to set n*48...
        adc     r0
        add     (sp)+,r0

        add     (sp)+,r1        ;Add n*2 to n*48 to set n*50...
        adc     r0
        add     (sp)+,r0

        return                  ;Return
.endc
.if df  bcd

; This routine goes through the list of clock register values and changes
; them from BCD to binary.

bcdbin:
        mov     #sec,r0         ;R0 => register value list
        mov     #10.,r1         ;R1 = repeat count
10$:    movb    (r0),r2         ;Get a value
        bic     #^C<360>,r2     ;Mask for 10's digit
        beq     20$             ;Nothing to do if zero
        ror     r2              ;Multiply this value by 10...
        mov     r2,r3
        ror     r2
        ror     r2
        add     r3,r2           ;Result in R2
        movb    (r0),r3         ;Get value again
        bic     #^C<17>,r3      ;Mask for one's digit
        add     r3,r2           ;Add this to previous result
        movb    r2,(r0)+        ;Store binary value
20$:    sob     r1,10$          ;Go do another until done
        return                  ;Then return
.endc

.end    setdt
```

RT-11 MACRO/FORTRAN Interactions

John M. Crowell
Los Alamos National Laboratory
Los Alamos, NM

Ned W. Rhodes, Session Chairperson
E-Systems
Falls Church, VA

Reported by Gavin Perry, DECUS Scribe Service

This tutorial covered the mechanism for calling MACRO routines from a FORTRAN program. The material is covered in the FORTRAN Users Guide and the FORTRAN Library Guide. When writing FORTRAN programs one sometimes needs to speed up certain critical parts of the program. These time critical routines will be faster if coded in MACRO routines which can be called from a FORTRAN program using the techniques presented here. These techniques include basic information on FORTRAN conventions for passing arguments to a subroutine. Also included are some of the pitfalls encountered when writing FORTRAN callable routines, with hints on how to get around them.

When a CALL statement is issued from a FORTRAN program, the code generated declares the subroutine name as a global and passes the address of an argument block in R5 to the routine as follows:

```
.GLOBL SUBRTN
       MOV   #ARG,R5
       JSR   PC, SUBRTN
ARG:   3  ; the number of arguments
       X  ; the address of parameter X
       Y  ; etc for the rest of the arguments
       Z
```

The MACRO routine can now get the argument addresses through R5. The first word of the argument block contains the number of arguments being passed in the low byte. The high byte, while usually 0, is officially undefined. This permits the use of certain tricks. (see below) The first word in the argument block is followed by the addresses of each of the arguments in the In the subroutine, these parameters may be accessed by indexing R5 (e.g. 2(R5)). This is safer than altering the value of R5, since other routines may also want to reference the arguments pointed to by R5.

Function Calls

A FORTRAN function call returns with the answer in R0. If the answer is Integer*4 or Literal*4 the low order result will be in R0 and the high order part in R1. For a Real variable, the low order portion of the result is in R1 and the high order

portion is in R0. Double precision returns four words with R0 containing the most significant portion of the result and the remaining portions in R1 to R3 (least significant). For a complex number the high order portion of the real number is in R0, the low part in R1, while the imaginary portions of the value will be in R2 and R3. FORTRAN expects to find the results returned to it in this format.

GOTCHA's

Missing arguments will have -1 as the address, so be sure to check for addresses of -1 when there is a chance of missing arguments in the call (eg CALL (A,B,,X)). If no arguments are passed the first word will contain 0 in the low byte, so you may want to check for that too. FORTRAN doesn't care if you save the resisters R0 to R5 and it won't save them for you between calls to your routines, so be sure to save any values that will be needed in other calls to a routine. The stack must also be saved. It is very important that for every push onto the stack there is also a pull from the stack. Four out of seven gotcha's were MIND THE STACK. It doesn't matter how many times it's said, everyone sometime ends up leaving a number on the stack. When this happens, a return to PC goes to never-never land. If the stack contains 0 the program will just exit without even saying good bye. Some pushes onto the stack are not obvious. For example, if CSISPC is called it pushes the number of switches onto the stack even if it is zero. Don't modify FORTRAN constants unless you want 2+2=5. It is not the value that is passed but the address. FORTRAN won't know that the value of the constant has been changed. If you are using the floating point instruction set and changing defaults for the precision or the mode, be sure to save the value of the current flags and pop them when done with the different mode or precision. Don't mung R5 until done with all the arguments or you may grab the wrong value.

FILE I/O

FORTRAN has an OTS work area where it keeps track of what I/O channels are open and various program linkages. Tell FORTRAN if you open or close I/O channels by using the library routines IGETC and IFREE respectively. Don't use CSIGEN since it closes channels 0-8 which are often opened by FORTRAN. Use CSISPC instead, but watch the switch number push on the stack. If FORTRAN doesn't know about the channels you open it may set a channel already open when it trys to open a channel.

COMMON BLOCK

The common block statement creates a PSECT which you can use from your MACRO routines; Just declare that PSECT in the MACRO. See the FORTRAN Users Guide for the format.

FORTRAN ERROR TRAPS

An error in FORTRAN causes a TRAP instruction with the argument 200 + the error number. You can use these errors to tell the user about fatal errors; be sure to do something that will allow for a graceful exit anyway (such as MOV -1, R0) since

a CALL SETERR may have been executed which will keep it from exiting until the error count reaches the count level specified. If there is any chance that error traps will be called (either you call them or you use OTS routines that may trap) be sure that the trap vector has been initialized.

The PSECT layout of a FORTRAN program was discussed. The first statement of a FORTRAN program starts with a call to $$OTI to initialize the OTS followed by a pointer to the MAIN which then points to the data to be initialized. It is possible to write threaded code to be used with threaded (OTS) routines and programs but it was not recommended. The threaded structure is a list of entry points to the threaded routines, followed by addresses for the parameters and constants. A naming convention identifies the FORTRAN operation codes the data types and address codes (to tell how many levels of pointers back to go before you'll reach the value). More information on this is available in the documentation of FORTRAN or from John Crowell.

MACDBG/RT-11: A User's Critique

John M. Crowell
Los Alamos National Laboratory
Los Alamos, NM

Reported by Margaret Watters, DECUS Scribe Service

John Crowell discussed the problems and the advantages of DIGITAL's debugging system, MACDBG. This program is a remote symbolic debugging tool which runs under RT-11SJ or RT-11XM on a PDP-11 or an LSI-11. This system has several features including the following: it loads programs into the target program via a Serial Line Unit (SLU); it examines and changes the target memory and registers; it has a RUN/HALT program; it is programmed to find breakpoints, watchpoints, and tracepoints; the Host console can be used as a target terminal; and it requires an ODT in the target. Finding the tracepoints is an especially useful tool in debugging, as the potentially problematic point in the target program is indicated, yet the program continues to run, so the user can observe the effect the point has on the program.

This system also has some non-essential features that are helpful. There is a "Help" page; a Status display (on the VT100 only); a command key pad (on the VT100 and the LA120 only); programmable keys (VT100 and LA120 only); indirect command files; and a logging console I/O to file.

The Debug Service Module (DSM) is optional, however it is required for finding breakpoints, watchpoints, tracepoints, and for single stepping. The DSM resides in the Target RAM, and takes up 464 octal bytes. It contains a loader for moving blocks of data into target memory. There is also a handler for handling breakpoints, watchpoints, tracepoints, and for single stepping. The module speeds up loading and depositing in the target memory. This module should not be used if the user does

not have the necessary RAM, or if he does not have RAM at locations 14 or 16. The program also should not be used if uses BPTs or if it used instructions that alter the T-bit. The DSM may be linked with the user's programs, and it can be loaded separately. The speaker warned that a user must be careful that

his program does not write over the DSM. He also warned that under SJ, the "halt" instruction corrupts DSM (or at least MACDBG thinks it does).

Crowell pointed out several problems that he has noticed while using MACDBG. The major problem is that MACDBG sets the cursor keys to the "Application Code", which does not matter while MACDBG is in use, since it does not use the cursor keys. However, MACDBG does not reset the keys to "Cursor Code" before exiting. This particularly baffled Crowell. MACDBG has several bugs of its own, but Crowell said that it is a powerful tool nonetheless, and that it is the most cost effective software that he has bought in a very long time.

Creation and Handling of Multi-Volume Directories Under RT-11 With TECO

Maarten van Swaay
Kansas State University, Department of Computer Science
Manhattan, KS

Jack Crowell, Session Chairperson
Los Alamos National Laboratory
Los Alamos, NM

Reported by J. Rick Mihalevich, DECUS Scribe Service

Many small RT-11 systems are based on floppy disk storage. Files and their backup copies can easily extend over 50-100 disk volumes, and locating an individual file can become a tedious chore of browsing through a large collection of disks or printed directories.

TECO can retrieve volume ID records and file names from a volume without invoking directory operations from USR. This capability makes it possible to use TECO for the creation of a multi-volume directory file. Because retrieval of the directory information does not invoke USR, the output volume can share a spindle with the input volumes from which the directories must be obtained.

A set of TECO macros for creation and use of a multi-volume directory file was described. The package presented includes provisions for the creation of a new directory file, for insertion / replacement / deletion of a single-volume directory in the file, and for locating selected files from the directory. Because the directory file can extend over more than half the space on a single volume, a mechanism was discussed to edit a large file-in-place.

The presenter offered copies of these macros. To obtain a copy one needs to send a floppy to: Maarten Van Swaay, Kansas State University, Manhattan, Kansas 66506. The presenter requested that a package complete with return address and enough postage for return be included with the floppy.

SOFTWARE PERFORMANCE REPORTS

OPERATING SYSTEM	VERSION	SYSTEM PROGRAM OR DOCUMENT TITLE	VERSION OR DOCUMENT PART NO.	DATE
RT-11	5.0	SIPP/PIP/BACKUP	V05.00	29-AUG

NAME: Ned W. Rhodes
FIRM: E-Systems, Melpar Division
ADDRESS: 7700 Arlington Blvd.
CUST. NO.: Falls Church, Va. 22046

DEC OFFICE AND CONTACT PERSON: Lanham
DO YOU HAVE SOURCE: YES [ ] NO [ ]

REPORT TYPE/PRIORITY

[X]	PROBLEM/ERROR	1. [ ] HEAVY SYSTEM IMPACT
[ ]	SUGGESTED ENHANCEMENT	2. [ ] MODERATE SYSTEM IMP
[ ]	OTHER	3. [ ] MINOR SYSTEM IMPACT
		4. [X] NO SIGNIFICANT IMPACT
		5. [ ] DOCUMENTATION/SUGGES

SUBMITTED BY: Gary L. Fuller PHONE: (703) 560-5000 X2858

CAN THE PROBLEM BE REPRODUCED AT WILL? YES [X] NO [ ]

ATTACHMENTS: MAG TAPE [ ] FLOPPY DISKS [ ] LISTING [ ] DECTAPE [ ] OTHER:

COULD THIS SPR HAVE BEEN PREVENTED BY BETTER OR MORE DOCUMENTATION? PLEASE EXPLAIN IN PROVIDED SPACE BELOW. YES [ ] NO [ ]

CPU TYPE	SERIAL NO.	MEMORY SIZE	DISTRIBUTION MEDIUM	SYSTEM DEVICE	DO NOT PUBLISH
LSI-11/23	AB02254	128 K	RX-02	RL-01	[ ]

1. A problem with the SIPP utility occurs when both of the following conditions are met:

   (1) An optional com-filespec is supplied in the SIPP command string.

   (2) A modification is made to the input file in the address range- 1000 $\leq$ (Base + Offset) $\leq$ 2000 (octal).

The problem is characterized by the insertion of the command file text (destined for the com-filespec channel) into the input file starting at address 1000 (i.e. Block 1).

The probable diagnosis is that there is a channel mix-up when buffering the command file text. Why the 1000 - 2000 address range is a factor is undetermined.

The problem may be reproduced by performing any of the customization patches supplied in the RT-11 Installation Guide (AA-H376B-TC) Ch. 2.7 which specify addresses in the indicated range and by additionally specifying a com-filespec.

2. The /H and /V switches are transposed in the minireference. They are correct in the system utilities manual.

3. BACKUP/MULTI - If a file is too large to fit on the output volume and it is the last file being transferred, it continues to prompt for output volumes instead of giving a message that the file is too large.

SYMPOSIUM TAPE INFORMATION

CALL FOR RT-11 SIG TAPE SUBMISSIONS

Assembling the RT-11 SIG Tapes at the DECUS Symposia (and producing a quality product) has turned out to be difficult. The Spring, 1983, tape was done after the Symposium, and I propose to do the same this time. Therefore, any SIG Tape submissions which are ready now can be sent to me for preparation. Please note, that even if you send a tape submission early, the DECUS U.S. Symposium Tape Copy Release Form MUST BE SIGNED!! A copy of the Release Form is attached below.

Please send all submittals, along with the Release Form to:

R. W. Barnard
Sandia National Laboratories
Division 2565A
P. O. Box 5800
Albuquerque, NM 87185

Remember that the RT-11 SIG accepts not only 9-track, 800 bpi, magnetic tapes, but also RX01 and RX02 floppies. (I can also read TU-58 DECtape II's). Thank you.

ATTACHMENT D

DECUS

Release Form Number:

TAPE COPY RELEASE AGREEMENT

The DECUS Program Library and the DECUS Tape Copy Facility provide a clearing house function only; programs are not sold or generated or tested. All programs and information and copies are provided "AS IS". DIGITAL EQUIPMENT COMPUTER USERS SOCIETY, DIGITAL EQUIPMENT CORPORATION, AND THE CONTRIBUTOR DISCLAIM ALL WARRANTIES ON THE PROGRAMS, INCLUDING WITHOUT LIMITATION, ALL IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS.

The following authorization is assumed for all programs copied on the copy facility:

Full permission and consent is hereby given to DECUS and to the DECUS Special Interest Group to reproduce, distribute, and publish and permit others to reproduce in whole or in part, in any form and without restriction, this program and any information relating thereto. The submitter hereby warrants and represents that he had good and sufficient right, interest, and title in and to this program and the related information to grant such permission to DECUS.

Signed _____ Date _____

ATTACHMENT C

Release Form Number:

DECUS U.S. SYMPOSIUM TAPE COPY RELEASE FORM

Name _____

Company _____

Address _____

_____

City _____ State _____ ZIP _____ Telephone _____

Program Name(s) _____

_____

SIG Tape Submitted to: ☐ RSTS/E ☐ VAX ☐ RT-11 ☐ TOPS-10
☐ RSX ☐ STRUCT. LANG. ☐ TOPS-20

Contents of Tape: _____

Number of Files _____ PPN _____

Is this material account specific? _____

Number and Kinds of Tape Submitted: ☐ DOS Format ☐ ANSI Format ☐ Other _____
☐ 7-track ☐ 9-track ☐ Other _____
☐ 800 BPI ☐ 1600 BPI ☐ Other _____

Description: _____

_____

_____

Guidelines:

Users who wish to participate in the exchange should bring a 2400 foot (preferably new) quality tape to the Symposium. The tape and cannister should be clearly labeled with the user's name and address.

1. ***No proprietary or licensed software (including whole or partial copies) may be submitted.***
2. Users who would like to submit modifications to licensed DIGITAL software may submit files to be appended to the original source program. ONLY the modifications may be submitted.
3. Users are encouraged to include a **README** file on their tape including the submitter's name and address, and a description of the files he/she is submitting.
4. Tapes should be compatible with standard system software. Please indicate the number of files and the PPN, UIC or account, and tape format.
5. Tapes should be 9-track, and be labeled with the sender's name and address.

IMPORTANT! RELEASE AGREEMENT ATTACHED